```python
#! python3.7
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, linalg, eye
from matplotlib.pyplot import style, figure, axes

def f(gamma) :
    u = ODESolving(a,b,N,u_left,gamma,eps)
    return u[N] - u_right

def DichotomyMethod(f,delta,N_max) :

    # Выделение памяти под массив значений gamma и значений функции f(gamma)
    gamma = zeros(N_max)
    function_value = zeros(N_max)

    # Задание начальных приближений для метода дихотомии
    gamma[0] = -10.; gamma[1] = 10.

    # Вычисление значений функции f(gamma) в точках,
    # определяющих начальные приближения для gamma
    function_value[0] = f(gamma[0])
    function_value[1] = f(gamma[1])

    # Реализация итерационного процесса метода дихотомии
    s = 1
    while abs(gamma[s] - gamma[s-1]) > delta :
        gamma[s+1] = (gamma[s] + gamma[s-1])/2
        function_value[s+1] = f(gamma[s+1])
        if function_value[s+1]*function_value[s-1] < 0 :
            gamma[s] = gamma[s-1]
            function_value[s] = function_value[s-1]
        elif function_value[s+1] == 0 :
            s = s + 1
            break
        s = s + 1

    return gamma[s]

def ODESolving(a,b,N,u_left,gamma,eps) :

    # Определение шага сетки
    h = (b - a)/N
    # Выделение памяти под массив сеточных значений решения ОДУ
    y = zeros((2,N + 1))
    # Задание начальных условий
    y[:,0] = [u_left, gamma]

    def f(y,eps) :
        f = zeros(2)
        f[0] = y[1]
        f[1] = 1/eps*(y[0]*y[1] + y[0])
        return f

    def f_y(y,eps) :
        f_y = zeros((2,2))
        f_y[0,0] = 0.; f_y[0,1] = 1.
        f_y[1,0] = 1/eps*(y[1] + 1); f_y[1,1] = 1/eps*y[0]
        return f_y

    # Реализация схемы из семейства ROS1
    # конкретная схема определяется коэффициентом alpha
    alpha = (1 + 1j)/2
    for n in range(N) :
        w_1 = linalg.solve(eye(2) - alpha*h*f_y(y[:,n],eps),f(y[:,n],eps))
        y[:,n + 1] = y[:,n] + h*w_1.real

    return y[0,:]

# Определение входных данных задачи
```

```
a = 0.; b = 1.; eps = 0.1
u_left = 4.; u_right = -3.5

# Определение числа интервалов сетки,
# на которой будет искаться приближённое решение
N = 50

# Точность, с которой ищется решение уравнения f(x) = 0 методом дихотомии
delta = 1e-12
# Максимальное число итераций в методе дихотомии
N_max = 50

# Поиск оптимального параметра стрельбы gamma с помощью метода дихотомии
gamma = DichotomyMethod(f,delta,N_max)
# Вычисление решения для найденного параметра gamma
u = ODESolving(a,b,N,u_left,gamma,eps)

# Отрисовка решения
style.use('dark_background')

fig1 = figure()
ax1 = axes(xlim=(a,b), ylim=(-6.,8.))
ax1.set_xlabel('x'); ax1.set_ylabel('y');
ax1.plot(linspace(a,b,N+1),u,'-ow',markersize=5)

# Листинг программы, реализущей приближённое решение краевой задачи
# для ОДУ второго порядка с помощью метода стрельбы
```